BYOP Project Report

2D to 3D Image Reconstruction

Domain: Computer Vision, Augmented Reality

by

Abhivansh Gupta
abhivansh_g@ch.iitr.ac.in
Chemical Engineering
Enr. No. 23112004
+91 8302905563
Mentors : Shree Singhi, Aakash Kumar Singh

Project Github Repo : https://github.com/avg16/byop-2425

INTRODUCTION

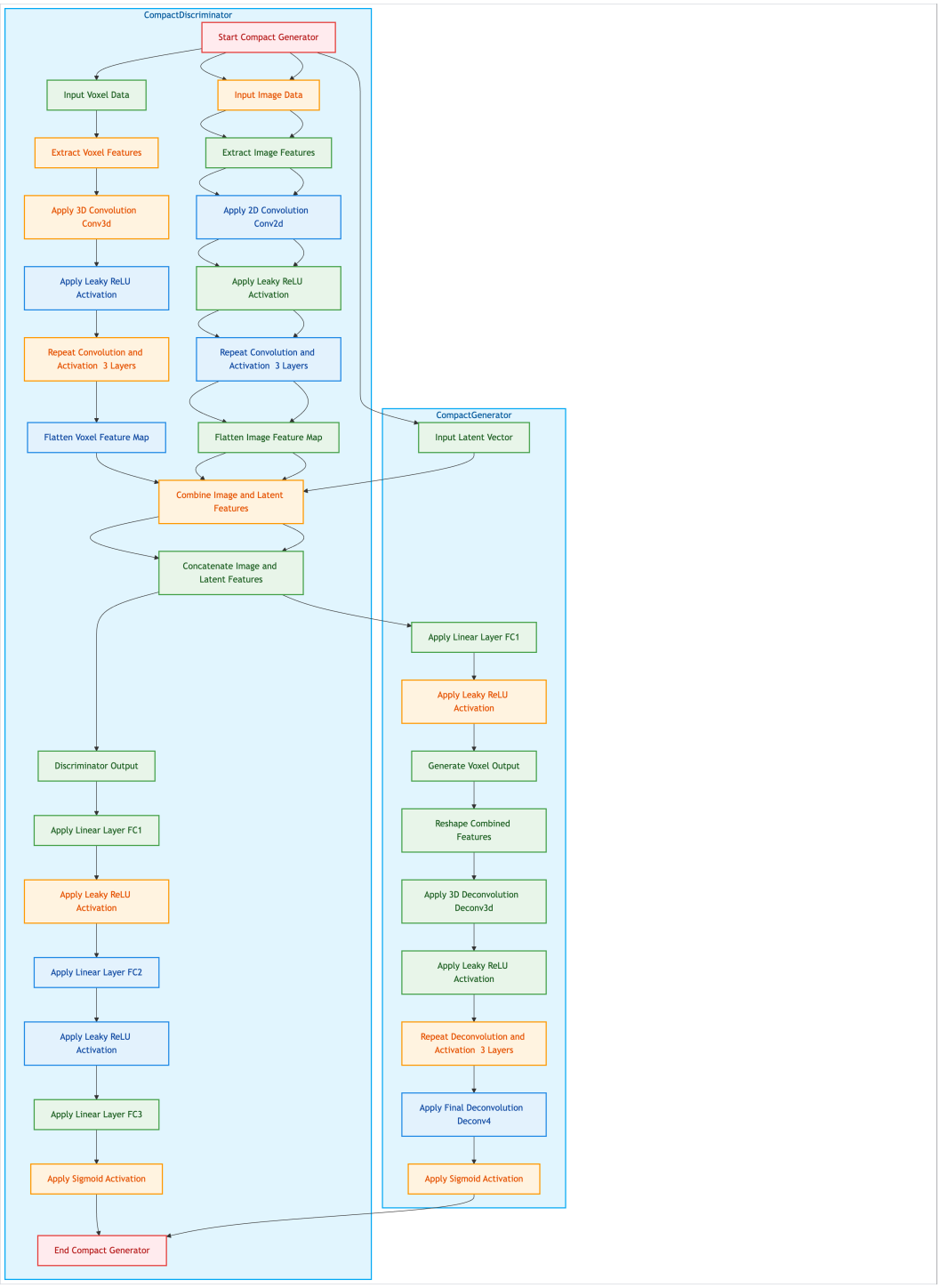
This project aims to address the challenge of reconstructing accurate 3D models using only 2D images. This work is about how Conditional Generative Adversarial Networks (cGANs) and Variational Autoencoders (VAEs) can automate and enhance 3D object reconstruction. By using the Pix3D dataset, which provides 2D images with corresponding 3D representations, the project seeks to build a generative model capable of creating 3D models.
The final output of the project is integrated for reconstructed 3D objects into AR environments, making them applicable to various domains. This not only makes 3D modeling more accessible and scalable but also demonstrates the potential of AI in bridging the gap between 2D data and 3D experiences.

APPROACH

Throughout this project duration, I constantly tried to figure out the best possible combinations and architectures for the task. The points/constraints I had to keep in mind were -
-The GPU accessibility

-Model compatibility
-To ensure the 3D data is effectively processed in required formats/extensions (voxel (.mat), .obj, .glb)
-AR deployment (via Homography)

So, discussing about the cGAN architecture, the flowchart below depicts the generator & discriminator used in the model:



So basically , in easy terms , in the **cGAN** structure, what I did :
In Generator architecture, used three 2D convolution layers, four 3D transposed Convolution layers, including a Linear. So ,2D convolutions here are extracting features from images, and 3D transposed convolution are constructing a 3D voxel grid. The model incorporates a latent vector ***z*** as well, that represents additional information to control the generation process. It is a conditional GAN just because 3D output is constructed using 2 inputs (the 2D image and the latent vector). If it was to be unconditional, only latent vector would be used. The latent vector is typically drawn from a prior distribution p(z), such as: $z \sim N(0,I)$ (a standard normal distribution).

This prior ensures that the latent space is well-structured and allows interpolation between points in the latent space to produce smooth transitions in the output. The latent vector captures these ambiguities by encoding additional information that is not explicitly present in the image. The hidden dimension in the architecture represents the number of feature maps and channels used in convolutions. However, increasing this parameter comes with the challenges of computation.

In Discriminator architecture, used three 3D voxel convolution layers, and three 2D image convolution layers. It is used to effectively differentiate between real and generated 3D voxel grids and their corresponding 2D images. The voxel convolution layers extract features from the 3D structures. The flattened voxel features and flattened image features are concatenated along the feature dimension. By combining these modalities, the discriminator can assess whether the voxel grid corresponds plausibly to the associated image. The fully connected layers used here project the concatenated features into a high-dimensional space. The final layer then outputs a probability score (sigmoid) representing the likelihood that the input pair (voxel, image) is real. The Leaky ReLU activation function is used after each layer: Provides non-linearity, allowing the model to learn complex decision boundaries. Avoids the "dying ReLU" problem by allowing small gradients for negative inputs, ensuring stable training. By keeping the number of channels and layers relatively small (e.g., starting hidden_dim = 16), the model maintains a compact architecture.

Coming to the **Variational auto encoder** , I prepared the complete workflow for it's training but I could not do it because even 5 epochs required a time of 6-7 hours on Colab's T4 GPU. This posed a challenge. Though discussing about the architecture used: consisted of two blocks, the Encoder & Decoder.

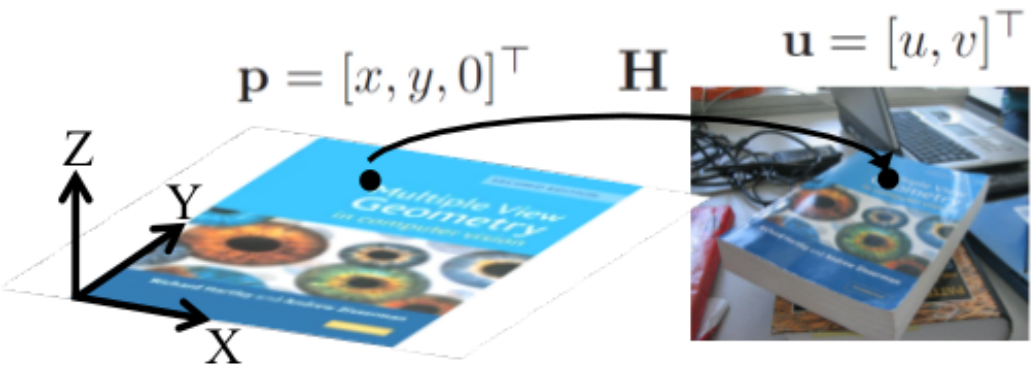
The Encoder architecture involved, computation of Mean and Variance. The encoder outputs mean (μ) and log-variance (log σ^2) which define a probabilistic latent space. The latent representation is sampled from a Gaussian distribution parameterized by μ and log σ^2. After the convolutional layers, the feature map is flattened to a vector of size batch size × flatten size.

These outputs are then used in the reparameterization trick : $z = \mu + \sigma \cdot \epsilon$

It allows the latent representation z to remain differentiable, enabling the encoder to be trained via back propagation despite the sampling step. The flattened size tells dimensionality of the learned features: It's a bottleneck that determines how much information is passed to the latent space.

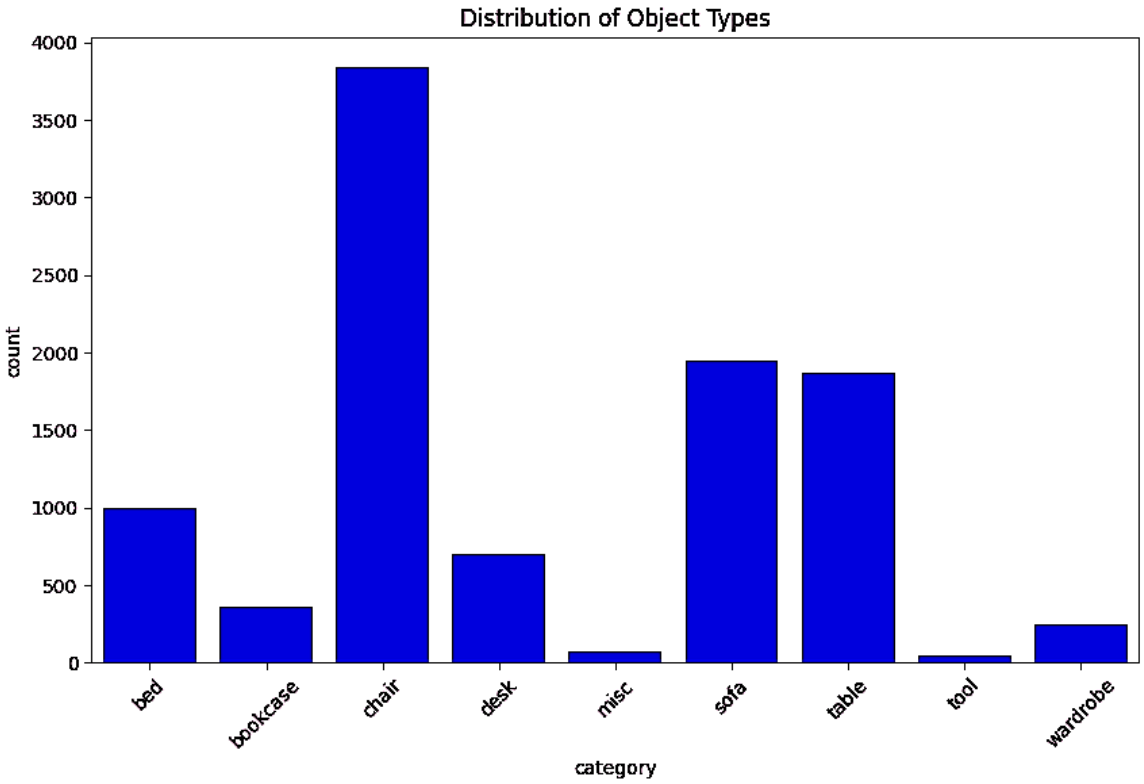
The Decoder architecture involved, 3D transposed convolutions, to generate the 3D structure back after being processed.

Coming to the Augmented Reality part, I didn't go through much of the Homography technique used, but referred to (n.wikipedia.org/wiki/Homography_(computer_vision)) this wikipedia article. In this technique, basically once camera resectioning has been done from an estimated homography matrix, this information may be used for navigation, or to insert models of 3D objects into an image or video, so that they are rendered with the correct perspective and appear to have been part of the original scene.



DATASET

Now coming to the training part, the dataset which I utilized for training is the Pix3D dataset (http://pix3d.csail.mit.edu/data/pix3d.zip) The dataset consisted of 10000+ rows, having various categories of furnitures like, beds, chairs, wardrobes, sofa, etc. each having their own model types, 3D data points (voxel form), 2d Images, Meshes, Masked representations.
A distribution of all the classes is given below:



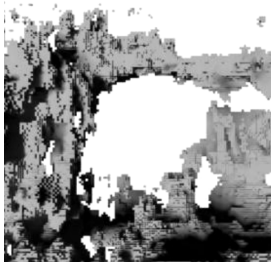
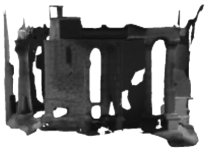
LEARNING OUTCOMES

Throughout the duration of this project, I did go through many articles, repositories & papers, constantly going through their approaches, architectures, the resources used, to create such scalable models and systems.
It was my first experience working with GANs, especially setting the pipeline to efficiently concatenate the 2D & 3D representations (took reference from papers, clarified via AI, as to why to do that). I also got to learn about the Homography technique (although a very basic idea of it). Articles I referred to :https://bitesofcode.wordpress.com/2017/09/12/augmented-reality-with-python-and-

opencv-part-1 , https://bitesofcode.wordpress.com/2017/09/12/augmented-reality-with-python-and-opencv-part-2
Overall, it was a good learning phase for me.

RESULTS & POTENTIAL IMPROVEMENTS

Below are some of the results that GANs could achieve:

Input	Output*	TRELLIS
		
		
		

* *the given outputs are the pictures of their .glb files, these files are interactive in actual. TRELLIS is Microsoft's 2D-3D reconstruction model. Sets a good benchmark.*

The results shown above do not cater perfectly to what one could think of at the start of the project. The possible reason for such results could be :

-Model Complexity (Overall, it was a simple model with not so much complex architecture involvement)

-Training issues (The model is trained on a small subset of the dataset as there were GPU constraints, and the process was consuming a lot of time which was limited during this time)

-Hyperparameter Tuning (Maybe, there were some other parameters which could help me generate better results. I tried different hyperparameter(s) and then finalized onto the existing one's.)

FUTURE SCOPE

I would love to contribute to this project after completing the BYOP phase as well, devoting more time in understanding the core (mathematics) behind the algorithm, seeking better implementation methods, refining the model more for better results to go for. Steps that can potentially improve this are:
Refinement in Reconstruction :

- Introduce Multi-View Supervised Learning to train the model using multiple images of an object from different perspectives.
- Use 3D Mesh Representation instead of voxel grids for smoother and more detailed reconstructions.

Loss Function Enhancements
Extend the project to use Text-to-3D methods by leveraging pre-trained CLIP models for joint text and image embeddings.

This project has a lot of utility in big systems today & tomorrow. Will love to contribute and learn more in the future.

ACKNOWLEDGEMENT: I would like to thank my mentors for providing constant guidance throughout the project duration.